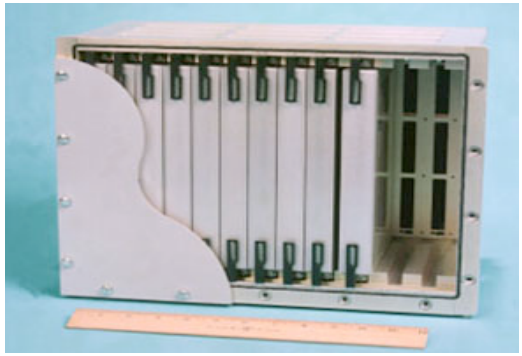
# Towards a New Software Stack for Extreme Scale Computing



***TeraScale***
Embedded

***PetaScale***
Departmental

***ExaScale***
Data Center


**Vivek Sarkar**
**Rice University**
vsarkar@rice.edu



**SC08 Workshop on "Power Efficiency and the Path to Exascale Computing", November 16, 2008**

# Acknowledgment: DARPA Exascale Software Study participants (work in progress)


- Saman Amarasinghe (MIT)
- Dan Campbell (Georgia Tech)
- Bill Carlson (IDA)
- Andrew Chien (Intel)
- Bill Dally (Stanford)
- Mootaz Elnozahy (IBM)
- Mary Hall (U. Utah)
- Robert Harrison (ORNL)
- Bill Harrod (DARPA)
- Kerry Hill (AFRL)
- Jon Hiller (STA)
- Norman Jouppi (HP)
- Sherman Karp (STA)
- Charles Koelbel (Rice)
- David Koester (MITRE)
- Peter Kogge (Notre Dame)
- John Levesque (Cray)
- Daniel Reed (Microsft)
- Mark Richards (Georgia Tech)
- Vivek Sarkar (Rice)
- Al Scarpelli (AFRL)
- Robert Schreiber (HP)
- John Shalf (LBL)
- Allan Snavely (SDSC)
- Thomas Sterling (LSU)

# Extreme Scale Systems

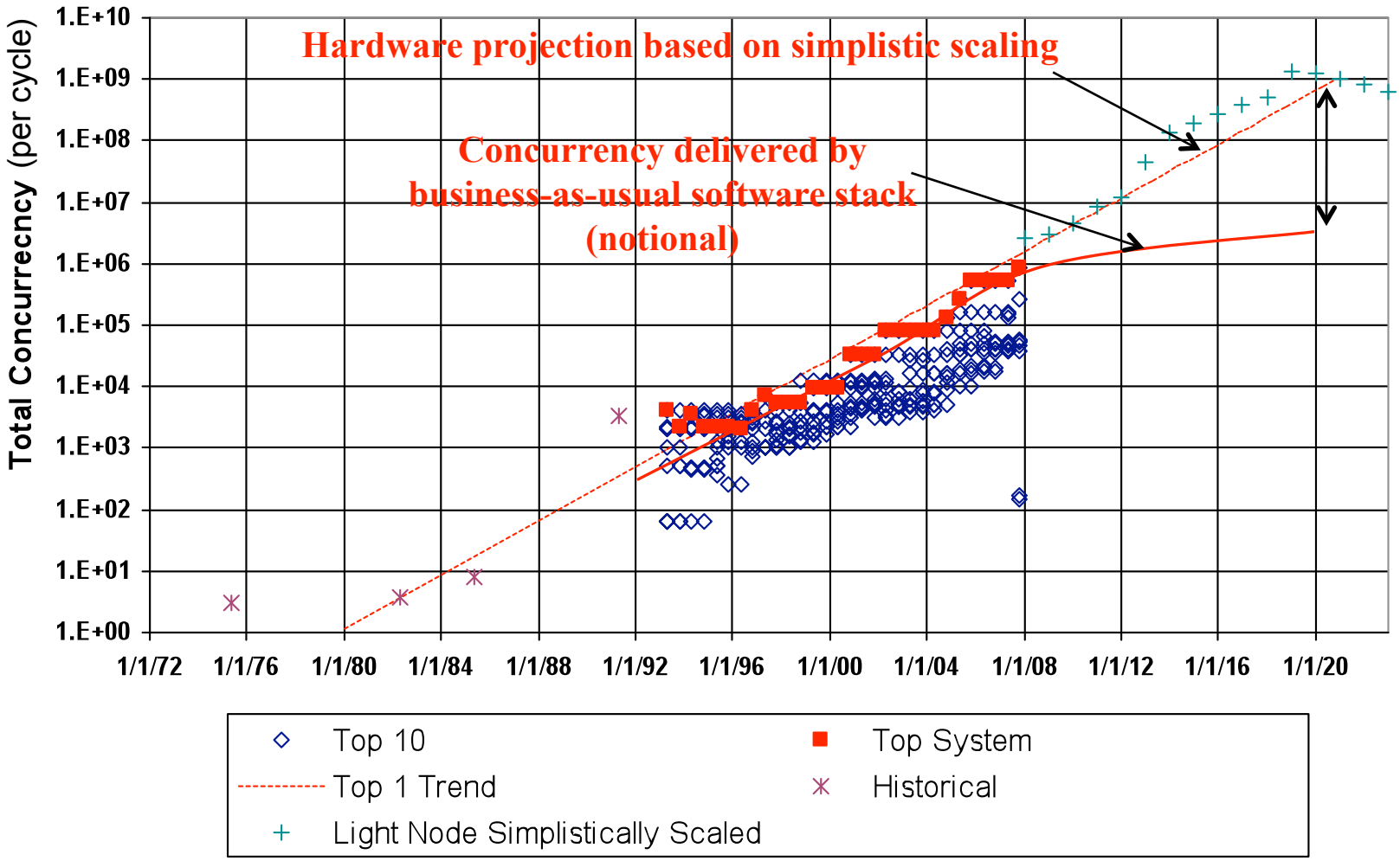
- **Characteristics of Extreme Scale systems:**
  - Massive multi-core (~ 1000 cores/chip)
  - Performance driven by parallelism, constrained by energy
  - Three system classes --- Exascale Data Center, Petascale Departmental, Terascale Embedded
- **Key Software Challenges:**
  - ***Concurrency***
  - ***Energy Management***
  - ***Resilience***
- **Software stack:**
  - Application frameworks & Tools
  - Programming models and languages
  - Libraries
  - Compilers
  - Runtimes for scheduling, memory management, communication, performance monitoring, power management, resilience
  - Operating Systems

*Productivity of upper levels depends on scalability of lower levels*

*Focus of this talk*

# The Exascale Software Challenge

***How to bridge the gap between Simplistically Scaled projections and reality of application software?***




**"ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems", P.Kogge et al**

# Challenges in Management of Parallelism and Locality

**OS Challenges:**

- **Scalability**
- **Locality**
- **Resource Management as Spatial Partitioning instead of Time Slicing & Interrupts**
- **Scalable access to Device Drivers; User Mode access of Messaging Hardware Resources instead of via OS**

**Run time Challenges**

- **Locality-sensitive Task Scheduling**
- **Fine-grained Communications**
- **Combining Synchronization with Dynamic Parallelism**
- **Memory Management**

# Challenges in Management of Parallelism and Locality (contd)

**Compiler Challenges**

- **Extracting Useful Parallelism from Ideal Parallelism**
- **Optimizations for Vertical and Horizontal Locality**
- **Synchronization Optimizations**
- **Communication Optimizations**
- **Energy Optimizations**
- **Global Auto-tuning and Dynamic Optimization**

**Library Challenges:**

- **Asynchronous scheduling of library tasks**
- **Communication libraries for fine-grained asynchronous data transfers**
- **Data Management and I/O Libraries for Extreme Scale**

# Concurrency Challenge

- **How will 1000x increase in concurrency be delivered to Extreme Scale Systems?**
  - Weak Scaling by increase in data size, $W_S$
  - Weak Scaling by increase in computation per datum, $W_T$
  - Strong Scaling by increasing parallelism in fixed computation, S

- **Over Provisioning required for Latency Hiding, L**
  - L is determined by latency hiding requirements --- memory access time, fraction of memory operations (parameterized by memory hierarchy level
  - Assume L ~ 100x

- **How much additional parallelism must come from software, relative to today's applications?**
  - ***$W_S$*$W_T$*S needs to be > $10^5$***

# Concurrency Challenge (contd)

- **$W_S$ is limited by DRAM cost and memory-computation ratio**
  - Petascale ~ 1 byte/FLOPS
  - Exascale ~ 0.01 bytes/FLOPS ??
  - Improving memory-computation balance is expensive
    - 50-50 cost balance dictates 0.004 bytes/FLOPS
    - 90-10 cost balance dictates 0.04 bytes/FLOPS
  - Assume $W_S$ ~ 10x

- **$W_T$ is limited by application domain and algorithm**
  - Assume $W_T$ ~ 10x

**➔ Software must deliver S ~ 1000x increase in Strong Scaling to use full capability of Extreme Scale systems**

# Concurrency Challenge: Implications on Software Stack

- **Fine-grained Parallelism**
  - Need to reduce task granularity by S ~ 1000x
  - ➔Task overhead in software stack needs to be reduced by 1000x
- **Fine-grained Synchronization**
  - S-fold increase in number of synchronizations performed
  - ➔ Synchronization overhead in software stack needs to be reduced by 1000x
- **Fine-grained Communication**
  - For 3D codes, surface area decreases by $S^{2/3}$ ~ 100x, as task size decreases by factor of S ~ 1000x
  - ➔ Communication overhead needs to be reduced by 100x
- **Scalability**
  - Amdahl's Law limit is reduced by a factor of S ~ 1000x
  - ➔ Sequential bottlenecks (critical path) in software stack needs to be reduced by 1000x

# Challenges in Application Development

- **Explore strong scaling and "new-era" weak scaling approaches for Extreme Scale applications e.g.,**
  - Multi-scale
  - Multi-physics (multi-models), Coupled models, New models
  - Mitigation analysis
  - Data-derived models
- **Explore new application areas for Extreme Scale**
  - Data mining e.g., Smith-Waterman
  - Real-time surveillance e.g., real-time analysis of 1 million sensors on a Departmental Extreme Scale system
- **Domain-specific Application Frameworks for Extreme Scale**
  - Extend approaches based on global arrays to irregular data structures (sparse tree, hash table) with fine-grained asynchronous parallelism
  - Integrate virtualization-based approaches (e.g., NAMD) with SPMD models such as MPI
  - Interoperability of frameworks for different domains in multi-physics applications

# Energy Management Challenge

- **Data movement and memory accesses will be most significant contributors to energy consumption in Exascale Systems (among contributors under software control)**
- **Locality challenges**
  - Spatial locality
  - Temporal locality
  - Data motion metrics and optimization
- **Other Energy Challenges for Software Stack**
  - Dynamic Voltage & Frequency Scaling
  - Power management of individual cores
  - Power management for Real-time deadlines
  - Reduction of Thermal Stresses and Hot Spots

# Combined metric for Concurrency & Energy

- **Combined figure of merit for concurrency, energy, resilience**
  - (Total energy) * (Elapsed time)
  - Analogous to energy-delay product
  - Above terms must include overheads for resilience support

- **C-A-S-H metric: The cost of an execution of application A on an Extreme Scale platform with system software stack S and hardware H can be expressed as**
  - C(A, S, H) = TotalEnergy(A, S, H) * TotalElapsedTime(A, S, H)
  - Use to compare $S_{old}$ vs. $S_{new}$ for same (A,H) pairs
  - TotalEnergy will be driven by localilty metrics
  - TotalElapsedTime will be driven by concurrency metrics

# Individual Metrics for Concurrency & Energy

## Concurrency:

- Thread level concurrency – active and pending (number of threads)
- Thread efficiency – fraction of total threads that are active at any one time doing useful work
- Synchronization overhead (clock cycles) and scalability (clock cycles as a function of number of threads)
- Thread management overhead (clock cycles)

## Energy:

- Energy to solution
- Performance per Joule (Op/J, Flop/J, Bytes transferred/J, energy oriented microbenchmarks)
- Locality metrics (data motion)

# Towards a New Software Stack for Extreme Scale Systems

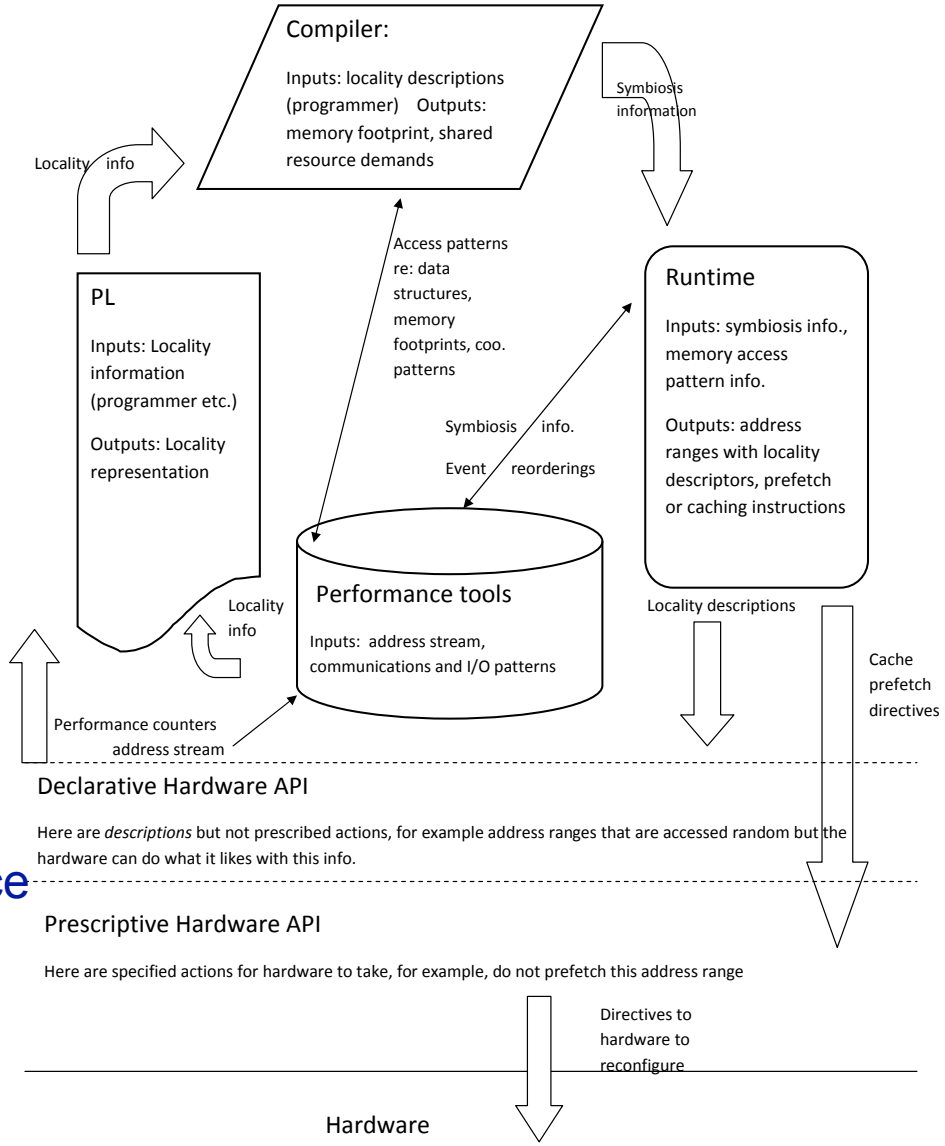
- **New system software "stack" that has the following features:**
  - Focus on extreme scale concurrency and locality
  - Focus on energy aware scalable techniques for resilience
  - Exploit and handle new trends in technology e.g., heterogeneity, many-core
  - Addition of energy awareness in all metrics

- **Self awareness and machine learning based on real time data gathering and analysis, and learned history lessons**
  - Focus on energy, concurrency, and resiliency

- **Will need long-term research that goes beyond industry focus on cloud computing**

# Software Stack and Hardware Interface

Software Stack



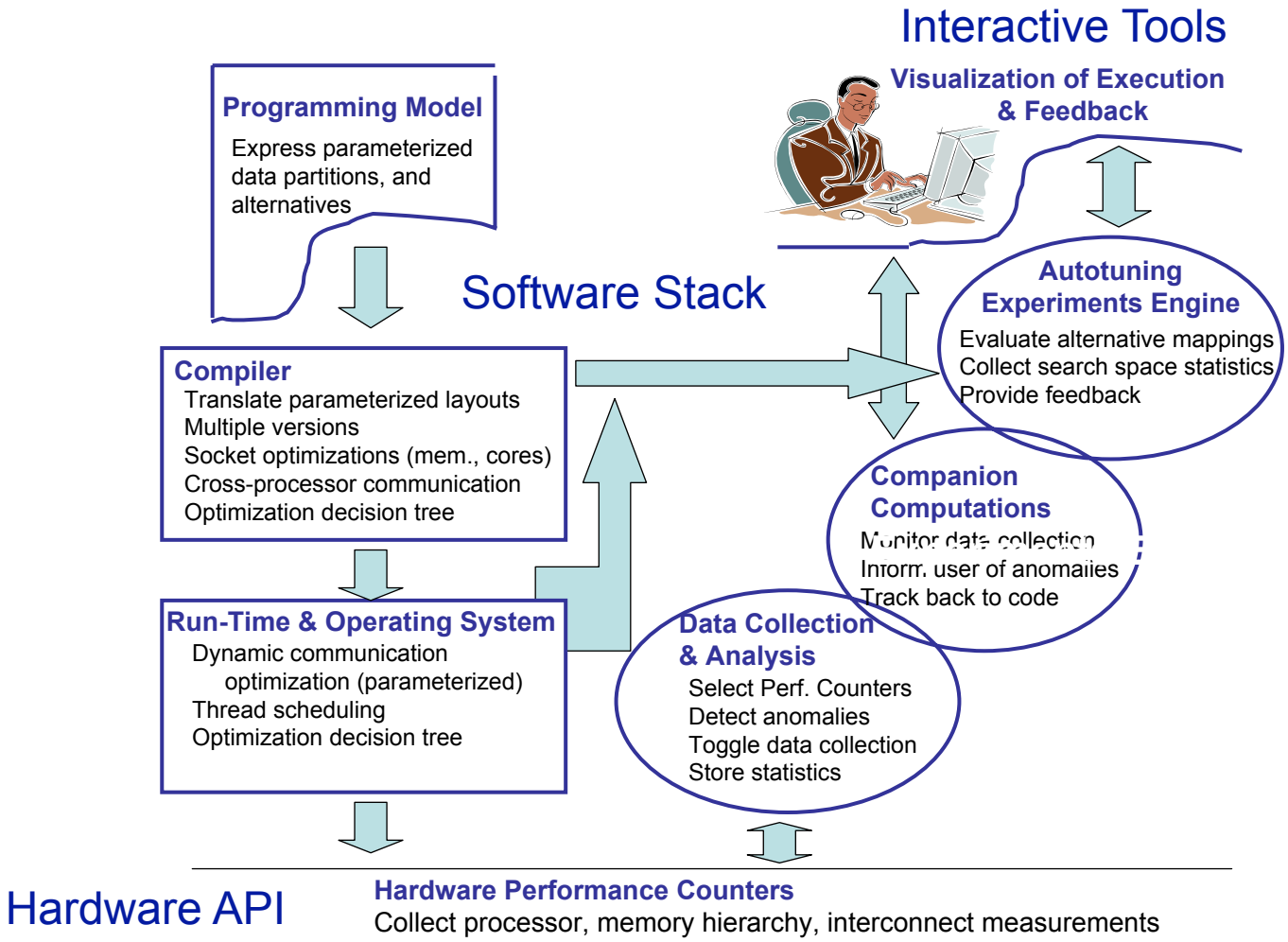

Hardware Interface

Specification of a "hardware interface" should drive hardware innovation in addressing requirements of Software Stack

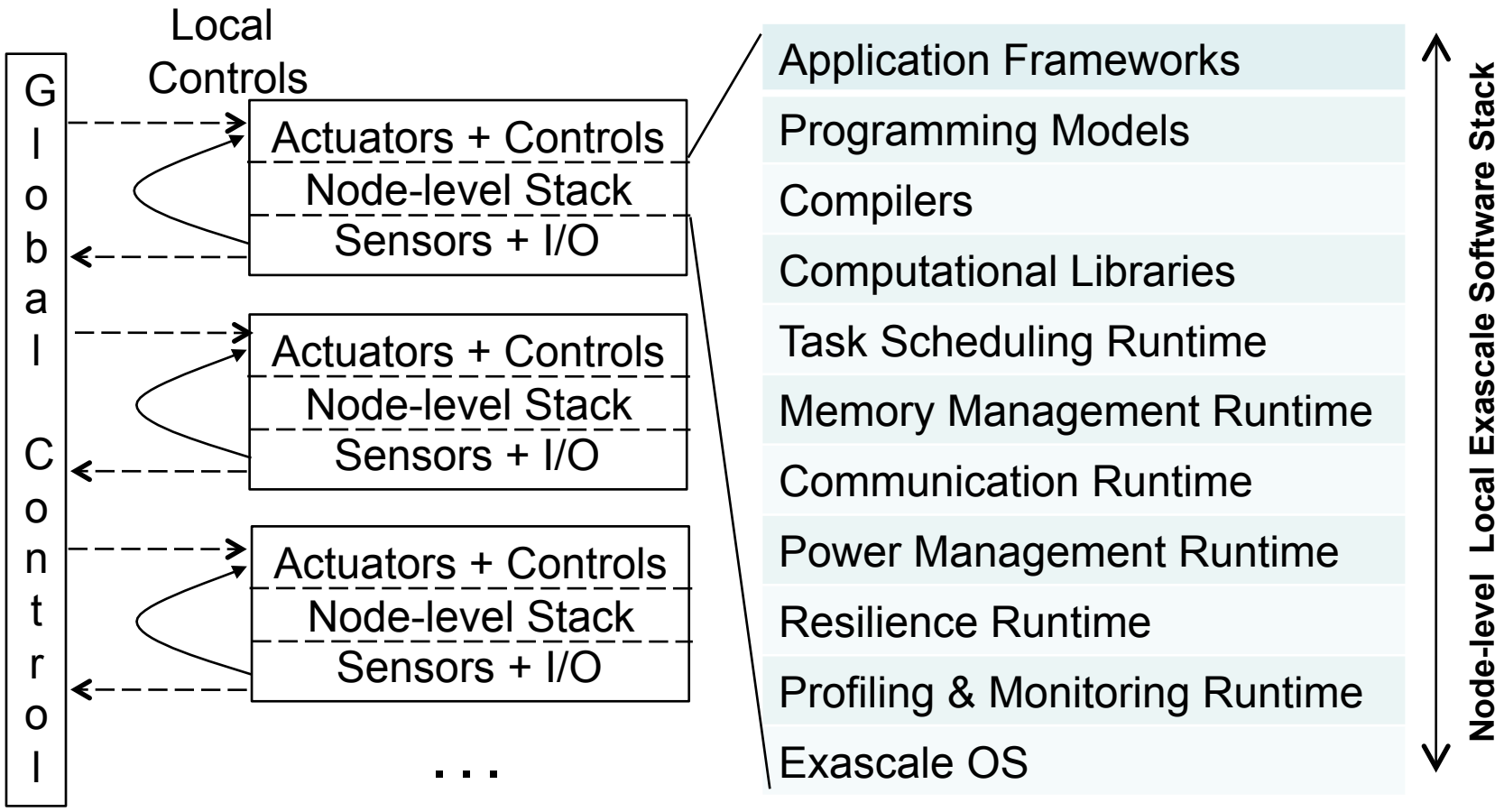
# Candidate items for Hardware Interface

- **Memory hierarchy configurations**
  - Cache sizes & geometries, hardware vs. software cache coherence
  - Register file sizes and data widths
- **Memory access patterns**
  - Address ranges that should bypass cache
  - Address ranges that require hardware coherence
  - Address ranges for which coherence will be managed by software
  - Address ranges with values that are guaranteed to be read-only (immutable) for certain application phases
- **Network bandwidth partitioning for different forms of data movement and communication**
  - PGAS, RDMA, Message passing, Stream processing, …
- **Other network reconfigurability parameters**
  - Topology, Packet size, …
- **Power management**
  - Frequency scaling, Voltage scaling, …
- **Performance profiling**
  - Lightweight profiling, Identification of events to be counted and sampled, …
- **Resilience**
  - Identification of threads with lower resilience requirements e.g., for which software can perform error detection and recovery

# Integrating Tools Requirements into the Software Stack



# Adding Global Self-Awareness to the Exascale Software Stack




# Why Now?

- **Hardware has changed dramatically while system software has remained stagnant**
- **Previous approaches have not looked at co-design of multiple levels in the system software stack (OS, runtime, compiler, libraries, application frameworks)**
- **Need to exploit new hardware trends (e.g., manycore, heterogeneity) that cannot be handled by existing software stack**
- **Emerging software technologies that address new software stack components exist, but have not been fully integrated with system software, nor addressed all Extreme Scale software requirements e.g.,**
  - Cilk, CUDA, Map Reduce
  - HPCS languages
  - Google file system
  - Autonomic computing

# Summary

- New research needed to build software stack for future Extreme Scale systems --- ExaScale Data Center, PetaScale Departmental, and TeraScale Embedded

- New software stack must support orders-of -magnitude increase in strongly-scaled parallelism, while staying within Extreme Scale Memory & Power Constraints

- New software stack necessary to realize advances in upper layers of software --- applications, programming models --- on Extreme Scale systems

# Come to our BOF!

- **Exascale Software Challenges Birds-of-a-Feather Session**
  - Allan Snavely, Vivek Sarkar
- **Tuesday, 05:30PM - 07:00PM, Room 19A/19B**
- **Abstract:** This BOF draws together experts and other interested parties focused on understanding challenges and developing promising approaches in developing robust, scalable, and efficient software to run at Exascale (nominally 1000x faster/larger/more concurrent than today's software). The session will begin with a high-level summary of an ongoing study on Exascale Software, and will be followed by 5-minute talks on topics that include: Exascale science drivers: what computations are amenable algorithmically to scale to 1000x? Coupling: Strategies for assembling coupled models from petascale components, what interfaces are needed to enable such "extreme coupling" to be efficient? Metrics: what metrics such as spatial and temporal locality, parallelism, and asynchrony are important to consider in Exascale software development? Compiler and runtime: what design principles are required to enable robust, efficient, scalable Exascale code generation and execution? Operating system: how should runtime support be partitioned between an Exascale Thin OS (ETOS) and user-mode system services?